

Л8. Периферійний послідовний інтерфейс UART

У Ардуіно реалізована апаратна підтримка інтерфейсу послідовної передачі даних через порти 0, 1. Периферійний послідовний порта UART мікроконтролера ATmega 328 використовується для обміну інформацією з BT, WiFi модулем, ультразвуковим датчиком HC-SR04, ємносним сенсором відбитків пальців Waveshare, датчиком CO2 K-33.

Його основні можливості: широкий діапазон швидкостей обміну даними; висока швидкість передачі при низькій частоті XTAL; 8 або 9-розрядний формат даних; виявлення помилок утрати даних при прийомі; виявлення помилок формату кадрів; виявлення помилкового стартового біта; три окремих переривання: по завершенню передачі, по порожньому регістру передавача і по завершенню прийому.

При передачі модуль UART додає до вхідного символу (8 або 9 біт) на початку — старт-біт (нуль), а в кінці — стоп-біт (одиниця), формуючи таким чином 10- або 11-бітову послідовність. Отримані значення передаються до регістра зсуву, який по черзі передає біти на вихід передавача TXD (вивід PD1). Швидкість видачі біт на вихід передавача визначається параметром *baud rate* (швидкість передачі інформації; вимірюється в бодах), яким можна керувати.

Приймач модуля UART безперервно перевіряє стан входу RXD, на якому за відсутності даних встановлюється рівень «1». Приймач зчитує інформацію з входу в 16 разів швидше. При виявленні на виводі RXD рівня «0» (тобто можливого старт-біта) мікроконтролер пропускає шість відліків, а потім робить три вибірки. Ці вибірки доводяться на відлік 8, 9 і 10 для кожного біта, що приймається, і, таким чином, зчитування значення біта відбувається в середині інтервалу його передачі, що дозволяє працювати з сигналами, що мають фронти великої тривалості. Якщо мікроконтролер виявляє, що на виводі RXD все ще присутній рівень «0», тобто прийшов стар-біт, модуль UART переходить в робочий режим і починає зчитувати байт. Якщо ж на виводі RXD вже присутній рівень «1», вважається, що перший відлік був просто шумом, і модуль переходить до очікування коректного символу. Якщо приймач визначив, що прийшов дійсний символ, він починає брати по три відліки кожного біта в середині інтервалу його передачі. Якщо значення всіх трьох відліків біта не збігаються, то значення біта набуває рівним значенню двох однакових відліків. На завершення модуль зчитує вибірки, що відносяться до стоп-біту. Для того, щоб було вирішено про коректний прийом символу, принаймні, дві з цих вибірок мають дорівнювати одиниці. Інакше модуль вважає символ за невірно кадрований і реєструє помилку кадрювання (framing error).

Розглянемо проект керування 4 виконавчими пристроями по UART. Якщо надходить цифра «1» включається/виключається канал 1, цифра «2» – канал 2,

цифра «3» – канал 3, цифра «4» – канал 4. Відповідно будуть задіяні порти D8, D9, D10, D11 Arduino. Схема проекту у середовищі Proteus наведена на рис. 1

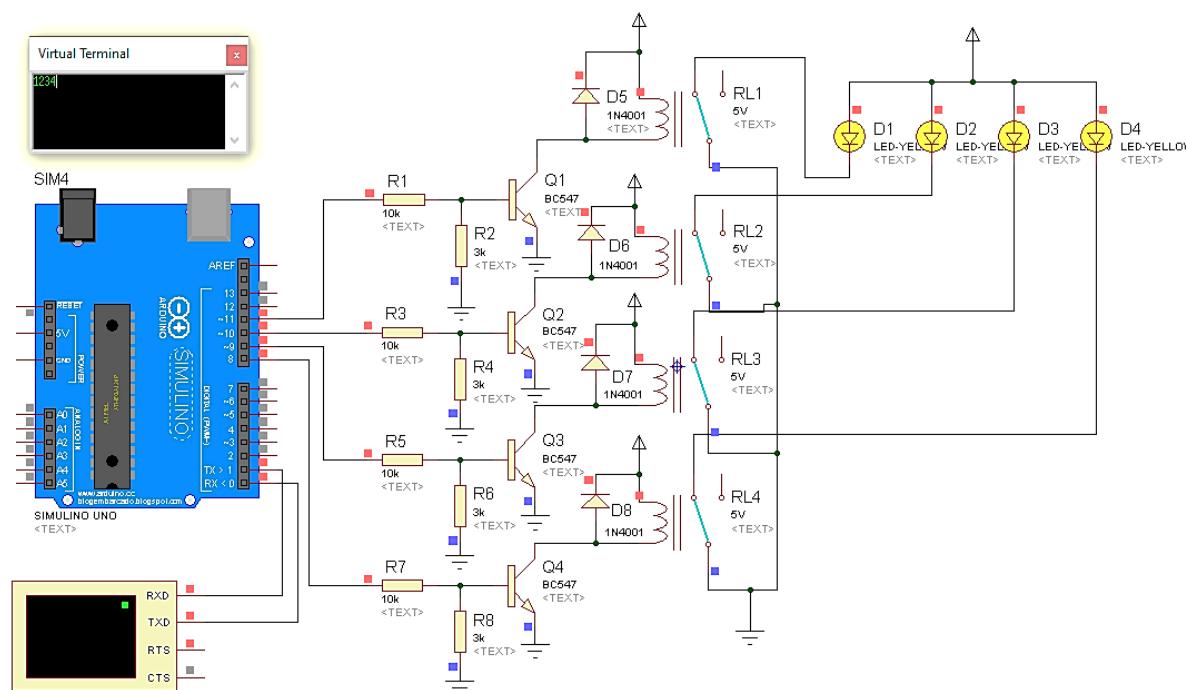


Рисунок 1 – Схема проекту керування 4 виконавчими пристроями по UART у середовищі Proteus

Лістинг програми (середовище Arduino IDE)

```
char incomingByte;
int b1=0;
int load1 = 8;
int load2 = 9;
int load3 = 10;
int load4 = 11;
void setup() {
    Serial.begin(9600);
    pinMode(load1, OUTPUT);
    pinMode(load2, OUTPUT);
    pinMode(load3, OUTPUT);
    pinMode(load4, OUTPUT);
}
void loop() {
    if (Serial.available() > 0) { //якщо є дані
        incomingByte = Serial.read(); // читаємо байт
        if ( incomingByte == '1') {PORTB^=0b00000001;}
        //переключити D8, решта залишити без змін
        if ( incomingByte == '2') {PORTB ^=0b00000010;}
        if ( incomingByte == '3') {PORTB ^=0b00000100;}
        if ( incomingByte == '4') {PORTB ^=0b00001000;}
    }
}
```

У реальному проєкті команди до Arduino передає модуль BlueTooth HC-06. Схема його підключення наведена на рис. 2 (HC06-Rx→Tx-Arduino, HC06-Tx→Rx-Arduino).

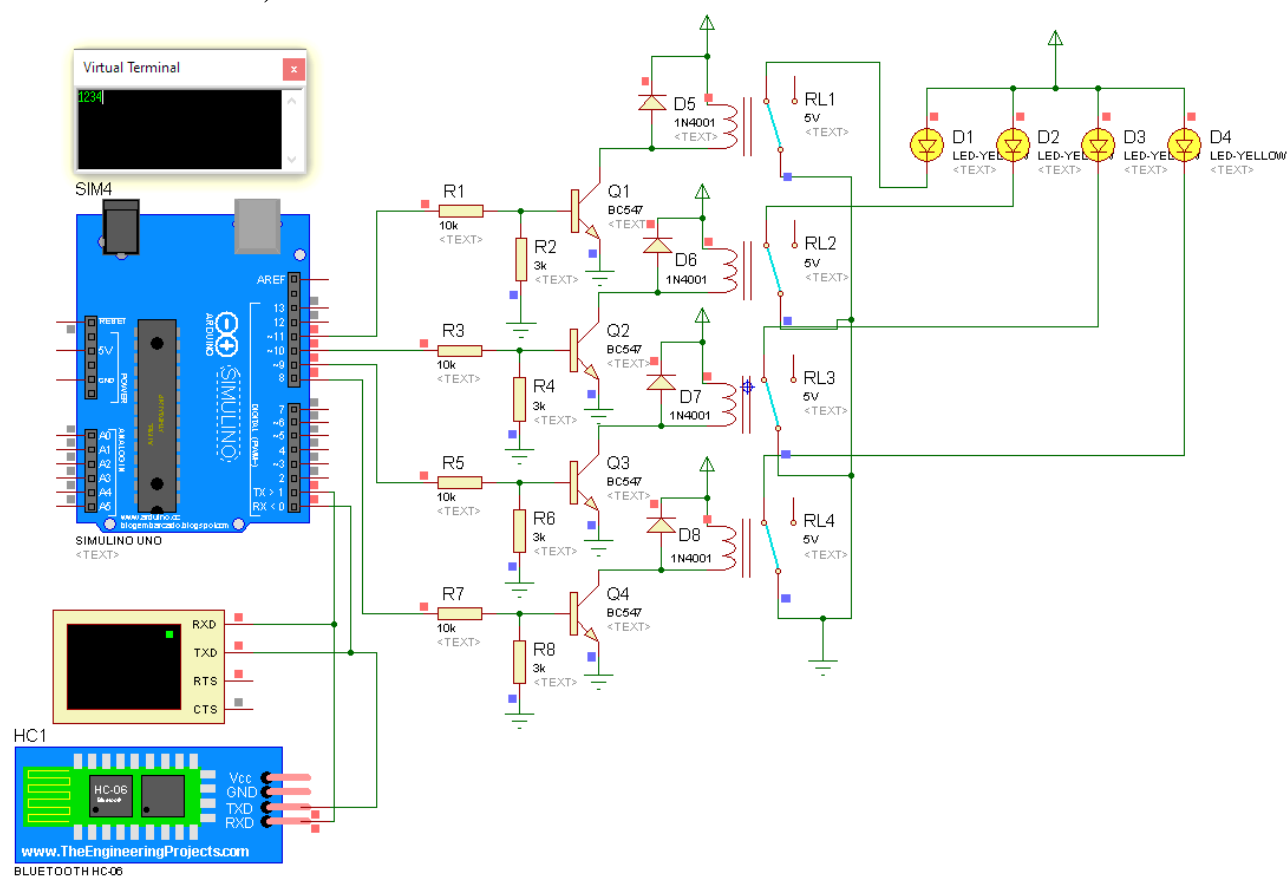


Рисунок 2 – Підключення BlueTooth HC-06 до Arduino

Розглянемо розробку мобільного додатку для керування 4 виконавчими пристроями через BlueTooth інтерфейс. Для цього переходимо до онлайн сервісу RemoteXY (рис. 3).



Рисунок 3 – Початковий екран онлайн сервісу RemoteXY

На рис. 4 наведений конструктор інтерфейсу мобільного додатку

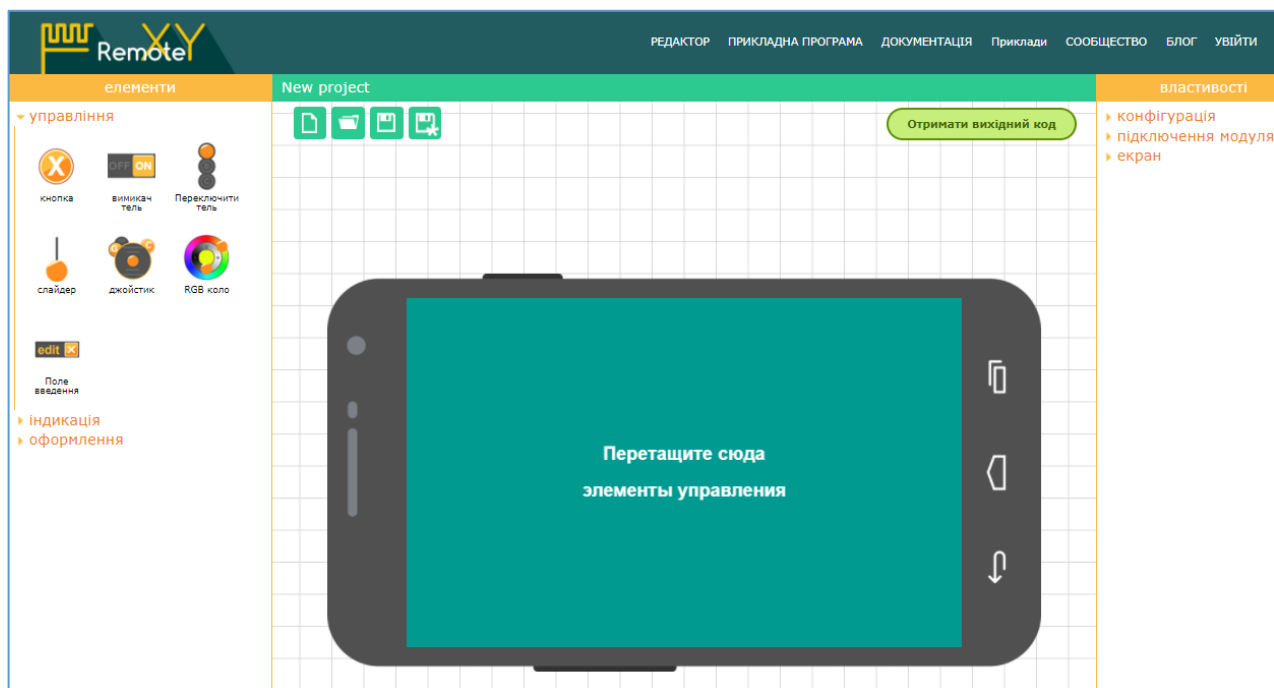


Рисунок 4. – Конструктор інтерфейсу мобільного додатку

Виконуємо конфігурацію проекту. Вибираємо тип Arduino, модуль HC06, середовище розробки Arduino IDE (рис. 5) .

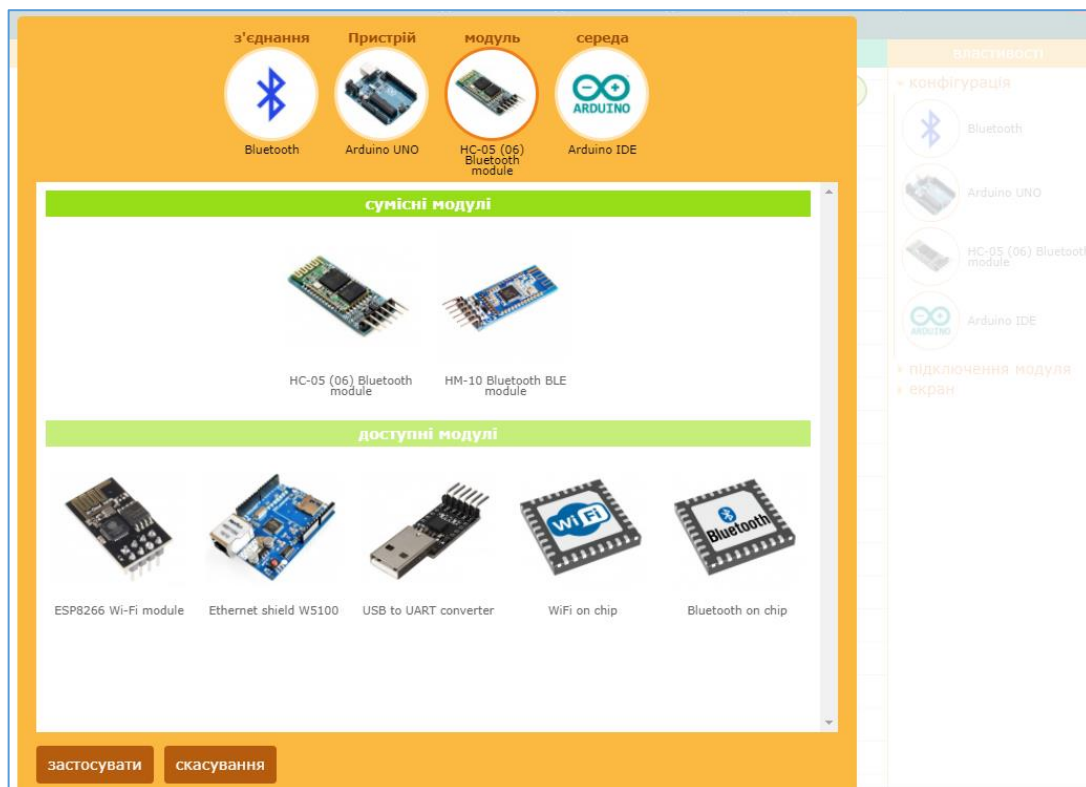


Рисунок 5 – Конфігурація проекту у середовищі RemoteXY

На рис. 6 показано налаштування підключення HC05 (Hardware Serial) та орієнтації екрану.

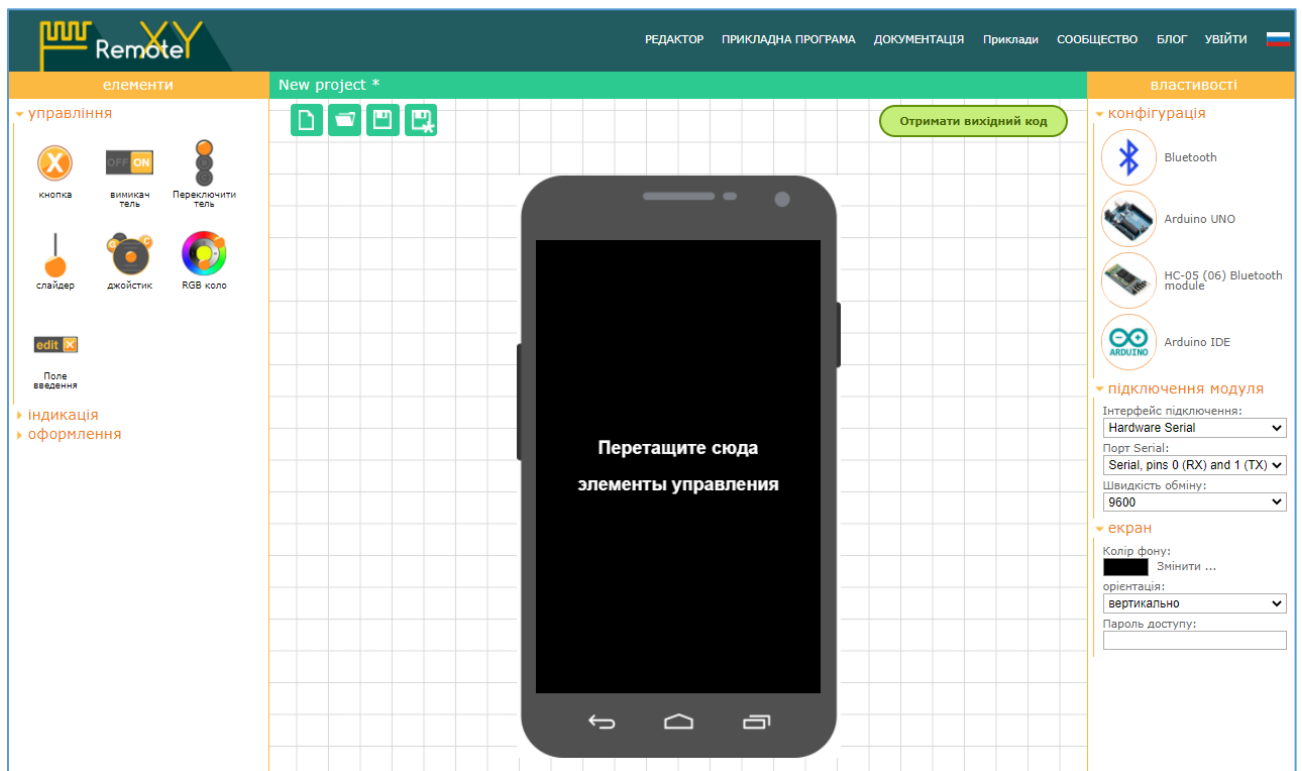


Рисунок 6 – Налаштування підключення HC05 (Hardware Serial) та орієнтації екрану



Рисунок 7 –Проектування дизайну мобільного додатку

На рис.7 показано проектування дизайну мобільного додатку. Встановлені 4 вимикача та 4 мітки. Для кожного вимикача виконується прив'язка до цифрових портів Arduino (D9, D9, D10, D11). Після цього натискаємо кнопку «Отримати вихідний код» та отримаємо інструкцію (рис. 8)

```
/*
-- New project --

This source code of graphical user interface
has been generated automatically by RemotexY editor.
To compile this code using RemotexY library 2.4.3 or later version
download by link http://remotexy.com/en/library/
To connect using RemotexY mobile app by link http://remotexy.com/en/download/
- for ANDROID 4.7.12 or later version;
- for iOS 1.4.7 or later version;
```

```

////////////////////////////////////
// RemoteXY include library //
////////////////////////////////////

// режим з'єднання та підключення бібліотеки RemoteXY
#define REMOTEXY_MODE__HARDSERIAL
#include <RemoteXY.h>

// настройки з'єднання
#define REMOTEXY_SERIAL Serial
#define REMOTEXY_SERIAL_SPEED 9600

// конфігурація інтерфейсу
#pragma pack(push, 1)
uint8_t RemoteXY_CONF[] =
{ 255,4,0,0,0,115,0,11,24,1,
  2,0,25,8,22,11,2,26,31,31,
  79,78,0,79,70,70,0,2,0,25,
  25,22,11,2,26,31,31,79,78,0,
  79,70,70,0,2,0,25,42,22,11,
  2,26,31,31,79,78,0,79,70,70,
  0,2,0,25,59,22,11,2,26,31,
  31,79,78,0,79,70,70,0,129,0,
  8,10,12,6,17,67,72,49,0,129,
  0,8,28,12,6,17,67,72,50,0,
  129,0,8,45,12,6,17,67,72,51,
  0,129,0,8,61,12,6,17,67,72,
  52,0 };

```

```

// структура визначає змінні та події інтерфейсу керування
struct {
    // input variables
    uint8_t switch_1;
    uint8_t switch_2;
    uint8_t switch_3;
    uint8_t switch_4;
    // other variable
    uint8_t connect_flag; } RemoteXY;
#pragma pack(pop)

////////////////////////////////////
//                               END RemoteXY include                               //
////////////////////////////////////

#define PIN_SWITCH_1 8
#define PIN_SWITCH_2 9
#define PIN_SWITCH_3 10
#define PIN_SWITCH_4 11

void setup() {
    RemoteXY_Init ();
    pinMode (PIN_SWITCH_1, OUTPUT);
    pinMode (PIN_SWITCH_2, OUTPUT);
    pinMode (PIN_SWITCH_3, OUTPUT);
    pinMode (PIN_SWITCH_4, OUTPUT);
    // TODO you setup code
}

void loop() {
    RemoteXY_Handler ();
    digitalWrite(PIN_SWITCH_1, (RemoteXY.switch_1==0)?LOW:HIGH);
    digitalWrite(PIN_SWITCH_2, (RemoteXY.switch_2==0)?LOW:HIGH);
    digitalWrite(PIN_SWITCH_3, (RemoteXY.switch_3==0)?LOW:HIGH);
    digitalWrite(PIN_SWITCH_4, (RemoteXY.switch_4==0)?LOW:HIGH);
}

```

Бібліотека **SoftwareSerial** дозволяє реалізувати послідовний інтерфейс на будь-яких цифрових портах **Arduino** за допомогою програмних засобів, які дублюють функціональність **UART**. Бібліотека дозволяє програмно створювати кілька послідовних портів, які працюють на швидкості до 115200 бод. Для пристроїв, що працюють з інвертованим сигналом, в бібліотеці передбачено відповідний параметр, що включає інвертування

```

#include <SoftwareSerial.h>
SoftwareSerial mySerial(10, 11); // RX, TX

void setup()
{
    // Ініціалізація послідовного інтерфейсу
    Serial.begin(57600);
    Serial.println("Good");
    // встановлюємо швидкість передачі даних для SoftwareSerial
}

```

```
    mySerial.begin(9600);  
    mySerial.println("Hello, world!");  
}  
  
void loop()  
{  
    if (mySerial.available()){  
        Serial.write(mySerial.read());  
        // TODO you setup code  
    }  
}
```